Neural-network design applied to protein-secondary-structure predictions

Richard C. Yu and Teresa Head-Gordon
Life Sciences Division, Lawrence Berkeley Laboratories, Berkeley, California 94720
(Received 21 November 1994)

The success of neural networks is often limited by a sparse database of training examples, deficient neural-network architectures, and nonglobal optimization of the network variables. The convolution of these three problems has curtailed the application of network models to protein-structure predictions, where homology modeling or information theory approaches are considered better controlled alternatives. This paper introduces our broad objective of disentangling the three degrading features of neural networks cited above, beginning with improved designs of network architectures used in the prediction of protein secondary structure. This work demonstrates that network architecture design considerations greatly improve generalization and more efficiently extract complex sequence-structure relationships from the existing database, as compared to arbitrary architectures with the same size input window.

PACS number(s): 87.10.+e

I. INTRODUCTION

A solution of the protein folding problem entails the discovery of the principles by which amino acid sequences encode information about, and efficiently find, their functional native states. A reasonable compromise of practical significance would be a straight structure-predictor algorithm, which would map sequences to structures with high accuracy while sacrificing a deeper understanding of forces that drive folding and eventually stabilize protein native states. Several such computational strategies exist, although none has demonstrated sufficient accuracy, and the reader is referred to several excellent reviews of the status of protein-structure predictions [1–3]. Statistical methods [4–7] and neural-network approaches [8–22] are united by their reliance on a database of known protein structures. It was originally thought that neural networks might exceed the predictive capacity of statistical methods that only exploit "first order" information, since hidden neurons might extract higher order correlations as well. However, neural-network approaches have not improved upon the best statistical analysis [6,7] or homology modeling [23] for secondary or tertiary protein-structure predictions. The attraction of neural-network methods is further diminished by empiricisms and technical difficulties needed to implement even a mildly successful prediction scheme [3].

The purpose of the neural-network models in the area of a secondary-structure prediction is to map relationships between certain input patterns of amino acids and the secondary structure of an element in that pattern. Three factors limit the capabilities of neural networks in this case (and in general). First, the performance of the network of an unknown input space is very dependent on how representative the trained database is. Relative to the number of proteins that have been sequenced biochemically (around 40 000 in sequence data banks such as SWISS-PROT and PIR), the number of solved structures is two orders of magnitude smaller—many of these homologous or structurally indistinct. This leads to a serious problem of under representation of correctly distributed information in the training examples for the network to learn. Second, the relationship between network topologies and performance is poorly understood, and thus network designs can be a hindrance to the practical predictive capabilities of the network, as well as being very difficult to interpret when examining final network weights. Finally, the choice of an optimal network training method is far from unambiguous: error backpropagation [24,25] cannot guarantee finding global error minima, while those that potentially have that capability [20,26,27] are terribly slow to converge. This problem is rooted in the complex, multidimensional error landscapes of the "multiple-layered" networks that are replete with local minima. The coupling of these three factors is even more problematic; it is difficult to address one problem singularity when the others are not completely understood and controlled.

The convolution of these debilitating features is well illustrated in the performance of neural networks used for a secondary-structure prediction. Rather simple networks using backpropagation and a finite input window predicted a test set with an overall accuracy of 63%, slightly higher if post-processed by cascading the output into another network, with positive correlation coefficients for each structural class [8,10,14]. Holley and Karplus found that adding more than two hidden units increased network memorization and Decreased network generalization. In fact, a network with no hidden units at all had almost the same prediction accuracy as the optimal, two-hidden-unit network [10,14]. Incorporating precalculated information about the periodicities of α-helix and β-sheet strands in the input representation for a one-hidden-layer, finite window size and three-state output network with postprocessing improved overall prediction accuracy (63% to 65%), and significantly improved correlations for helix, without sacrificing sheet or coil correlation [12].

Kneller, Cohen, and Langridge [12] also tried classify-

ing their protein database into tertiary structural classes based on types identified by Levitt and Chothia [28]; for networks trained over a particular structural class, significant overall prediction accuracy improvements are realized. Recent networks which exploit sequence homologies have now gained another 5% in prediction accuracy [11]. Together, these network results best display the need to address the problem of database sparsity and limitations of a finite input window.

To move beyond these pioneering attempts to predict secondary structure using neural networks will require projecting out and solving individually the three limitations to successful network application cited above. In this paper and future work, we consider strategies that decouple these degrading features of network approaches to provide for further improvements in network performance. Thus far we have considered network architecture design for both helix prediction of real proteins and tertiary structure for complete sequence-structure data bases of model chemistries [21,22,29]. This paper is devoted to a preliminary exploration of the neural-network topology problem for secondary-structure prediction for real protein databases using a finite input window size of nine amino acids. Section II presents relevant background material necessary for the subsequent sections. Our hand-designed network structures presented in Sec. III are, in effect, constraints, reducing the amount of information the network has to learn by "hard-wiring" rules already learned by other means. In Sec. IV we will show that these topologies positively influence network prediction and most significantly improve generalization to new sequence-structure relationships as compared to previous neural-network models. As summarized in Sec. V, methods for overcoming database sparsity will be necessary to realize significant improvements in predictive capacity, but our results suggest that careful attention to network design will strongly influence predicative confidence and the avoidance of undesired areas of configuration space of the network variables.

II. BACKGROUND AND METHODS

Network training. In this study we work exclusively with feed-forward backpropagation networks [30]. A typical multilayer network is shown in Fig. 1. Learning

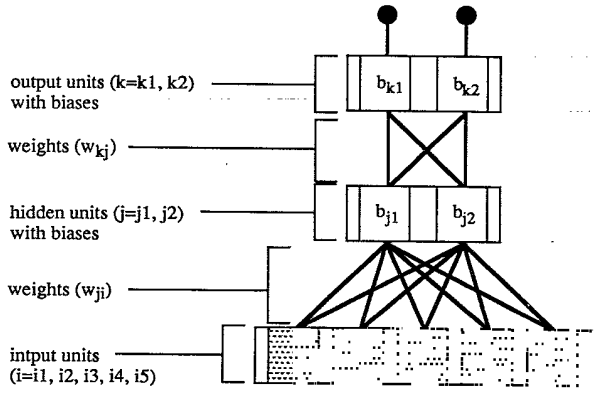

FIG. 1. A generic feed-forward, multilayer, neural-network architecture.

by backpropagation involves minimizing an error function that goes to zero as the final calculated outputs of the network match the observed output for all input-output pattern pairs. The function traditionally used is [30]

$$E=\sum_i \sum_j (O_{oj}^i - O_{ci}^i)^2 , \tag{2.1}$$

where the summation index i is over all patterns, summation index j is over all output units, O_{oj}^i is the observed (i.e., database value) output for output unit j and pattern i, and O_{ci}^i is the calculated (network) output, which is calculated as follows. An input pattern i is fed into input units, which output the values of the pattern elements. These values are fed forward through the network; all the network units calculate their outputs according to

$$O_j^i=\tanh(\beta A_j^i) , \tag{2.2}$$

where

$$A_j^i=\sum_k (w_{jk} O_k^i - b_j) , \tag{2.3}$$

and the summation index over k is over all connected downstream neurons. β is a steepness factor of the hyperbolic tangent function ($\beta=1.5$ in this study), w_{jk} is the weight connecting the downstream neuron k to the upstream neuron j, and b_j is the threshold bias associated with neuron j. In this paper, outputs are represented by real values from -1.0 to 1.0, and the nonlinear response function matching this output range is a hyperbolic tangent, unless otherwise noted.

Minimizing the error involves taking steps down the error gradient with respect to the free variables, the weights and biases:

$$\Delta w_{jk}=-\gamma \left[\frac{\partial E}{\partial w_{jk}} \right] , \tag{2.4}$$

$$\Delta b_j=-\gamma \left[\frac{\partial E}{\partial w_j} \right] , \tag{2.5}$$

where γ is generally some fixed step size (set at 0.001 in this paper). While the above shows the error as a sum over all patterns, we follow the common alternative method that computes and steps down a gradient for each pattern. By choosing the patterns randomly, we can step through weight space stochastically and search through a wider portion of the error surface [30]. The above process of calculating and stepping down the error gradient is repeated for each input-output pattern pair in the training set until a stopping condition is met (we consider the training complete after 500 passes through all network patterns). During training, we also randomly reject input-output patterns of a certain secondary structure class with a probability related to the class's representation in the training database. This heuristically attempts to address underrepresentation of helix and sheet relative to coil in the database, and we note that similar methods have been previously employed with success [20].

Network performance. After a network is trained, there are several methods of evaluating its performance.

A good review of these is found in Ref. [20] and they are briefly explained in this section. The simplest one is the fraction correct,

$$Q_s = \frac{P_s}{N_s} , \quad (2.6)$$

where P_s is the number of correct predictions of structure s, and N_s is the number of observed residues of structure s. The total correct Q_{tot} is just the sum of all correctly predicted residues normalized by the total number of observed residues. By focusing on criteria like Q_{tot}, we get networks that potentially predict more of overrepresented output categories (e.g., coil) at the expense of the other categories. Perhaps the most meaningful measurement of accuracy is the Matthews correlation coefficient:

$$M_s = \frac{(P_s R_s - U_s O_s)}{\sqrt{[(R_s + U_s)(R_s + O_s)(P_s + U_s)(P_s + O_s)]}} , \quad (2.7)$$

where R_S is the number of residues that are not in structure S that are correctly rejected, U_s is the number of residues underpredicted, and O_s is the number of residues overpredicted. This number accounts for overpredictions and underpredictions of the state in question; it is equal to 1.0 when the network predictions are perfectly correlated with the observed output, 0 if random, and -1.0 if there is anticorrelation. We assign a quality factor (QF) defined as the sum of the Matthews correlation coefficients as a one-value measure of a network's performance. By doing this, we place the emphasis on the network that best learns all three structural categories most accurately.

Secondary structure classification. The majority of neural-network secondary-structure prediction research has worked with a three-state structure classification—helix, sheet, and coil. There are many different criteria for assigning secondary structure; the dictionary of secondary structures of proteins (DSSP) method developed by Kabsch and Sander [31] is in widespread use, making comparisons between different prediction methods more meaningful. Various combinations of hydrogen bonds, clearly defined by a polar interaction energy cutoff, build simple elementary structures, such as turns (an N turn at residue i is defined as a bond between i and $i+N$) and bridges (between residues i and j) that are parallel (bonds between $[i-1,j]$ and $[j,i+1]$ or between $[j-1,i]$ and $[i,j+1]$) and antiparallel (bonds between $[i,j]$ and $[j,i]$ or between $[i-1,j+1]$ and $[j-1,i+1]$). These elementary structures form more complex structures—i.e., runs of N turns form helices, and consecutive bridges form ladders that in turn group to form sheets. The majority of neural-network studies that predict secondary structure consider residues in four-turn helices to be "helix," while resides in ladders are classified as "sheet," and everything else is designated as "coil." Our definitions are the same except that we consider isolated β bridges as sheet residues as well. In addition to the Kabsch-Sanders algorithm, a dictionary of secondary-structure clarifications for all amino acids for 62 proteins from 19 different families is published [32]. This paper uses these 62 portions, 48 grouped into a training set and 14 into a testing set as in Ref. [10]. We note that the DSSP secondary-structure assignments are not unique, but provide an algorithm which is very methodical and hierarchical based on hydrogen bonds. While details of the network design approach presented here will change with changing secondary-structure assignments, the modifications necessary would be straightforward to implement.

III. NEURAL-NETWORK DESIGN

Previous work [21,22] has shown that network topologies can be rationally designed to encode certain constraints or information abut the input; in the present case, we can "hard wire" certain knowledge about secondary-structure dependence of particular amino acids. For example, a simple five weight network topology can capture the intuitive essence of hydrophobic attraction, given the relative hydrophobicities of two particular amino acids as inputs, the output would represent an increased likelihood of proximity [22]. Other knowledges can be encoded in network topologies, much like the above function that models hydrophobic interactions (hydrophobic central Boolean function, or CBF) [22]. They come from a combination of physical forces, database analysis, and definitions imposed by the secondary-structure assignment algorithm. In the first category, both hydrophobic and electrostatic effects can be encoded in a CBF network topology (with different weights). In the second, statistical analysis of particular groups of window positions can also inspire network functions. Finally, constraints are also implied by the DSSP secondary-structure assignment algorithm. In addition, input representations can encode relative scales for helix and sheet propensity and hydrophobicity for all amino acids. The encoding of these types of information is described in the remainder of this section; in the following text, hydrophobic residues are abbreviated as H, neutral residues as O, and hydrophillic residues as P. Unless otherwise noted, (i) refers to the neural window residue.

Input representation. Input representation involves the conversion of the amino acid or padding residue to network values. In previous neural-network benchmarks [8,10], each residue is represented by a group of 21 bits (20 amino acids plus padding residue), where the bit position representing the particular amino acid is on (1) while all others are off (0). The weight from this bit is the only one modified by the backpropagation step. This representation purposely avoids encoding any knowledge of input-output relationships. In our study, instead of a 21-bit input representation, each of the input pattern's elements is converged to a three-value group (Fig. 2). A group consists of a helix propensity, a sheet propensity, and a hydrophobicity; each of those is from a scale normalized to values from -1.0 to 1.0. The hydrophobicity scale, derived from x-ray structures of globular proteins, is from Ref. [33]. The other scales were determined by examining the distribution of amino acid types of residues of a secondary structure from the training set. For each amino acid X, we determined its position on the helix propensity scale by normalizing the number of the

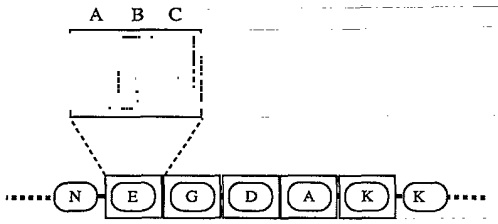

FIG. 2. The representation of amino acids in a finite input window. Each amino acid element is converted to a three-value group representing its relative helix propensity, sheet propensity, and hydrophobicity in discrete values of -1.0 (least), 0 (neutral), and 1.0 (most).

amino acid X in train set helix structures by the number of the amino acid X in the whole train set. The same was done for sheet propensity. Based on the just described procedure, the 20 amino acids are assigned a real number between -1.0 and 1.0 as a ranking for their relative helix propensity, sheet propensity, and hydrophobicity. The continuous ranking of amino acids for each scale is then divided into three discrete values of -1.0, 0.0, and 1.0.

Helix propensity. As stated above, the DSSP algorithm requires helices to be a minimum of four residues long. For the central residue i to be in helix, at least one of four possible blocks of four contiguous residues, $(i-3)$ to (i), $(i-2)$ to $(i+1)$, $(i-1)$ to $(i+2)$, or (i) to $(i+3)$, must be classified as completely helix. Then, if at least one of the four blocks is composed of residues with high helix propensity, then we can send a positive signal to the helix output unit. Each of the four groups has an associated subfunction [Fig. 3(a)] that sums up the helix propensities of the residues in the group; the bias of the summing unit determines how "prohelix" the sum of the residues' helix propensities must be before the subfunction fires positively. Each subfunction feeds into an integrating unit [Fig. 3(b)] with an internal bias determining how many of the subfunctions must fire positively to send a final, positive signal. (The integrator bias's initial value is set so that the unit replicates logical OR functionality; that is, it outputs 1.0 if any of the inputs is 1.0.)

Sheet propensity. A minimal sheet structure is two residues long. There is a greater likelihood of the central residue being a sheet when two contiguous residues containing position i consists of amino acids with high sheet-forming propensity $(i-1$ and i or i and $i+1)$. Similar to the helix propensity function, subfunctions are set up for the two contiguous residues [Fig. 4(a)], which are then fed into an integrating unit [Fig. 4(b)] whose initial bias value is set to make the unit mimic logical OR functionality.

Sheet hydrophobicity. Examination of the database revealed that i and $i+1$ pairs in sheet structures were more often comprised of *H-H* and *H-O* pairs, with *P-P*, *P-O*, *H-P*, and *O-O* pairs being disfavored. This logic is easily implemented by a single hidden unit [Fig. 5(a)]. We then encode two separate functions. If pairs containing the central window residue $(i-1,i)$ and $(i,i+1)$ are *P-P* or *P-O*, the inhibitory function fires and the excitory function remains neutral; if both pairs are *H-H* or *H-O*, the excitory function fires and the inhibitory function

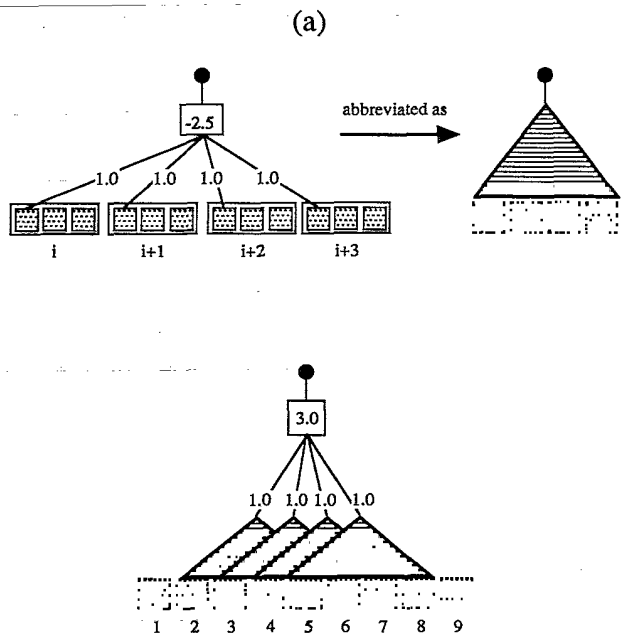

FIG. 3. Helix propensity subfunction. (a) Helix propensity input field for (i) to $(i+3)$ is connected to an integrating hidden unit with a bias of -2.5. (b) The combination of (a) for $(i-3)$ to (i), $(i-2)$ to $(i+1)$, $(i-1)$ to $(i+2)$, or (i) to $(i+3)$ provides a logical OR function, which fires positively if *any* subfield fires differs positively.

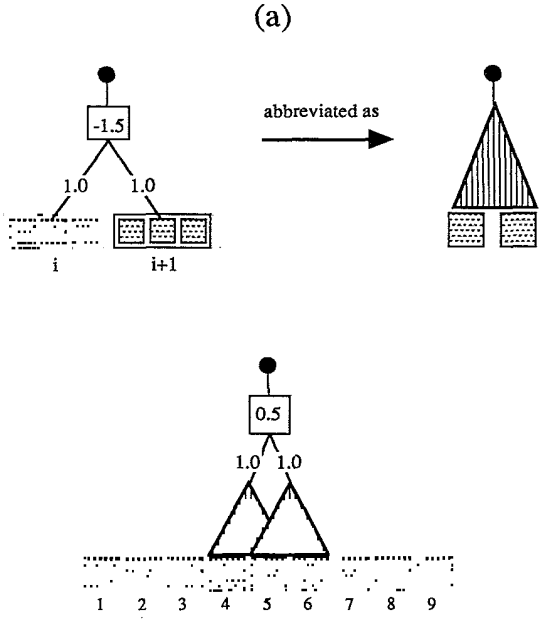

FIG. 4. Sheet propensity subfunction. (a) Sheet propensity input field for (i) and $(i+1)$ connected to an integrating hidden unit with -1.5 bias value. (b) The combination of (a) for $(i-1,i)$ and $(i,i+1)$ provides a logical OR function, which fires positively if *any* subfield fires positively.

remains neutral. This is accomplished by having two units with [0,1] output ranges [i.e., we use $\exp(bx)$ as a response function] connected to the pair of subfunctions as shown in Fig. 5(b). $(i,i+2)$, $(i,i+3)$, and $(i,i+4)$ pairs in sheet structures all revealed similar hydrophobicity correlations, and similar functions were designed for the related input positions.

Helix hydrophobicity. If the central residue is part of a helix, the DSSP algorithm demands that either $(i-3)$ to (i), $(i-2)$ to $(i+1)$, $(i-1)$ to $(i+2)$, or (i) to $(i+3)$ be all helix. If $(i-3)$ to (i) are helix, then there is a 4-turn at $i-4$ and at $i-3$, meaning that $(i-4,i)$ and $(i-3,i+1)$ are bonded. (There are similar results for the other possibilities.) In this case, we might expect the pairs $i-4$ and i, or $i-3$ and $i+1$, to both be hydrophobically similar (*H-H* or *P-P*). The CBF function in Fig. 6(a) is the basic building block of the first level of four subfunctions, each of which is an AND of central Boolean functions from two consecutive $(i,i+4)$ pairs. The final level in the function outputs positively if at least one of the AND's is true. If all possible cases are hydrophobically dissimilar, the final function output is inhibitory [Fig. 6(b)].

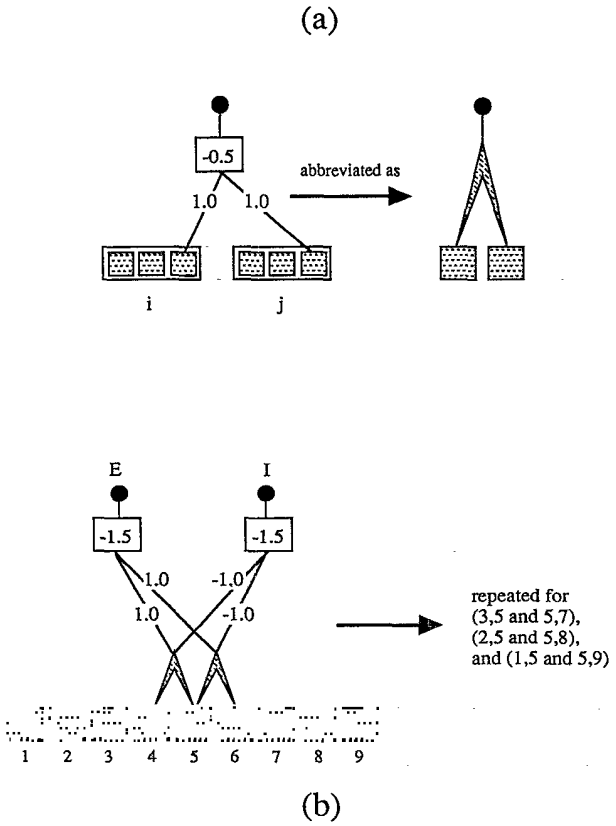

FIG. 5. Sheet hydrophobicity subfunction. (a) Hydrophobicity input field for (i) and (j) connected to an integrating hidden unit with -0.5 as value. (b) Combination of $(i-n,i)$ and $(i,i+n)$, for $n=1$, 2, 3, and 4. The overall structure has two outputs, one excitatory and the other inhibitory. The excitatory node fires positively and the inhibitory neutrally if the input pairs are *H-H* or *H-O*; the inhibitory node fires positively and the excitatory neutrally if the input pairs are *P-P* or *P-O*. See text for further explanation.

Analogous to hydrophobicity, (i) and $(i+n)$ pairs in helix structures contained fewer neutral (O) residues and more hydrophobic and hydrophillic residues than (i) and $(i+n)$ pairs in nonhelix residues. The basic logical function, with two inputs, outputs -1.0 if either of the inputs are O, and 1.0 of both inputs are non-O [Fig. 7(a) is shown]. This subfunction is similar to the sheet hydrophobicity case. If both subfunctions are positive, the output is prohelix; if both subfunctions are negative, the output is antihelix or else the output is neutrally 0 [Fig. 7(b)]. An excitatory integrating unit takes input from the logical subfunctions at $(i-n)$ and (i) and (i) and $(i+n)$, and outputs $+1.0$ if all subfunctions indicate non-O; likewise, an inhibitory integrating unit fires -1.0 if the subfunctions detect O-like residues. Like the sheet, there is both an excitatory and an inhibitory function for $n=2,3$ and 4 as well.

Output representation. One typical three-state output

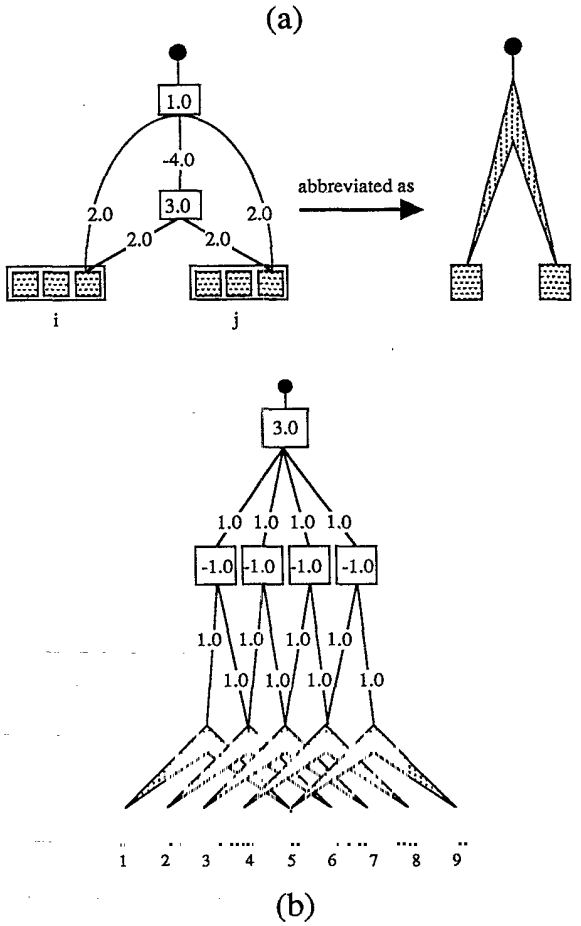

FIG. 6. Helix hydrophobicity subfunction. (a) Hydrophobicity input field for (i) and (j) are connected in the central Boolean function (CBF) topology described in [22], where pairs which are hydrophobically similar (*H-H* or *P-P*) fire positively while hydrophobic-hydrophillic pairs (*H-P*) are discouraged. (b) Four logical AND functions are defined (hidden units with -1.0 bias), which combine consecutive pairs $(i-4,i)$ and $(i-3,i+1)$, $(i-3,i+1)$ and $(i-2,i+2)$, etc. These are integrated into a logical OR function that fires positively if *any* of the logical AND's fire positively.

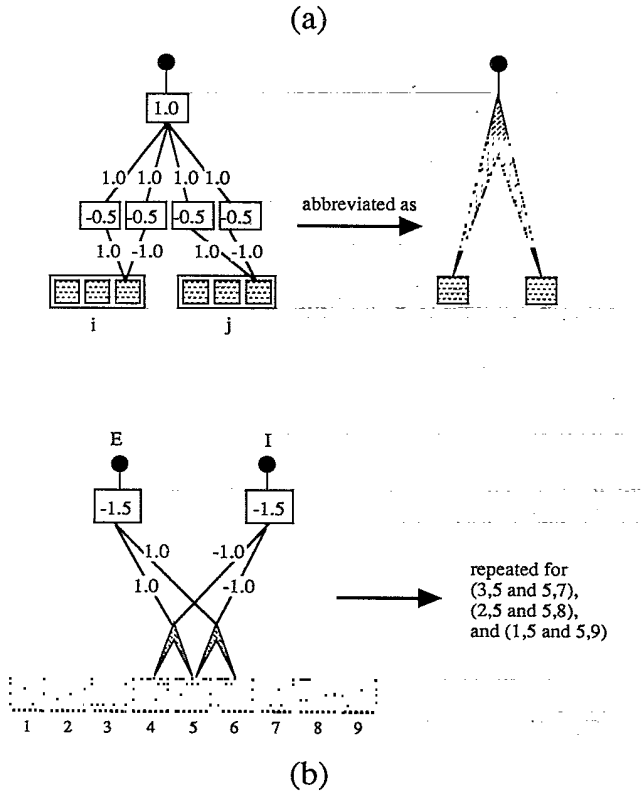

FIG. 7. Helix hydrophobicity subfunction. (a) Similar to sheet hydrophobicity subfunction [Figs. 5(a) and 5(b)].

representation (helix, sheet, coil) is a group of two values, one helix and one sheet [10]. This is implemented by comparing the two outputs against a threshold t; if either of the outputs is above t, then the structure associated with the higher output unit is the predicted structure; otherwise the network output is considered to be coil. The networks in the paper follow a modified version of this approach by allowing for two threshold parameters (one for each unit) searched for in steps of sizes 0.01.

IV. NEUTRAL-NETWORK PERFORMANCE

The results presented here are preliminary in nature, where input representation, window size, and structural homologies have not been fully explored and therefore optimized. In spite of these known limitations in the present hand-designed networks, they exhibit two very important benefits when compared to previous network results. First, the networks that are designed to describe secondary structure generalize to the test set naturally, so that the optimized network shows *generalization* superior to those with arbitrary topologies. Second, they provide an improved means of training by essentially serving as constraints in a constrained optimization of the network variables. The remainder of this section demonstrates these two points with several well-defined controls. Table I summarizes our results for all networks considered in this section, and all details of the networks discussed here (such as specific weights and biases) are available from the authors.

Our optimal hand-designed topology is shown in Fig. 8. The functions described in the previous section are fully connected to a hidden layer, which is then fully connected to the output layer. An additional hidden layer is fully connected to the input layer and the output layer, and we refer to them as "context" neurons. Both hidden layers consist of two hidden units; larger hidden unit layers were not explored. We designed the network (and all in this study) with input windows wide enough to contain our hand-designed topologies. The Kabsch and Sander helix hydrophobicity function and helix propensity function (see above) both extend from $i-4$ to $i+4$, so a window width of 9 is used. We justify the use of all the above described functions by assessing their performances individually. The train and test sets were modified so that the output patterns were changed to two state for a particularly designed function. For example, the helix functions were evaluated on the train and test databases with outputs of helix or nonhelix. All functions exhibited positive correlation coefficients, as expected.

The first set of controls demonstrates the positive influence of hand-designed elements on network generalization. We compare the optimal structured network design to simple two- and three-layer networks (Fig. 9) using the same input and output representation described in the previous section. Two networks of the form 9__11__2 (B1) (nine element input windows, 11 units in the first and only hidden layer, and two-state output) and 9__9__9__2 (B2) are considered, both with comparable

TABLE I. Secondary-structure predictive accuracy and Matthews coefficients for hand-designed and arbitrary topology networks with nine element input window.

	Train					Test				
Network	Q_{tot}	M_{helix}	M_{sheet}	M_{coil}	QF	Q_{tot}	M_{helix}	M_{sheet}	M_{coil}	QF
A	0.57	0.36	0.32	0.31	0.99	0.56	0.32	0.30	0.33	0.94
A[a]	0.59	0.37	0.33	0.29	0.98	0.58	0.34	0.28	0.31	0.93
A2	0.54	0.34	0.29	0.30	0.92	0.54	0.31	0.27	0.33	0.91
A3	0.56	0.37	0.33	0.31	1.00	0.56	0.33	0.30	0.33	0.96
A4	0.57	0.40	0.35	0.31	1.06	0.54	0.31	0.25	0.32	0.88
B1	0.56	0.40	0.38	0.26	1.04	0.47	0.20	0.20	0.21	0.61
B2	0.56	0.43	0.41	0.26	1.10	0.45	0.18	0.19	0.19	0.56
B3	0.56	0.40	0.33	0.26	0.99	0.50	0.25	0.25	0.23	0.73

[a]Refers to the Holley-Karplus method of postprocessing final network outputs to collapse all helix runs less than a length of 4, and sheet runs less than a length of 2, into coil.

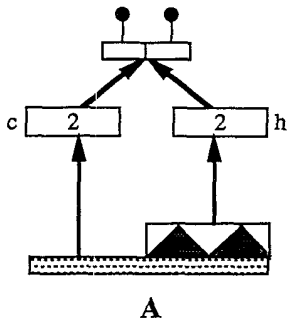

FIG. 8. The optimally designed network A. The network is composed of a combination of the hand-designed pieces described in Figs. 3–7 (and graphically described by solid triangles) connected to a two unit trainable hidden layer (denoted *h*), which in turn is fully connected to the two unit output; in addition, there are two units of "context" neurons fully connected to the input layer and the output layer (denoted *c*).

numbers of training weights to network A and together providing some variation on network topology. These more flexible networks (compared to the large percentage of fixed variables in the case of network A) perform significantly better on the training set as judged by the QF factor but perform much worse on the testing set. Clearly these topologies are largely memorizing the training set at the expense of test set generalization. Even for the 9__6__2 (B3), the best of two-layer networks in the sense that the test set prediction is maximized, network A shows superior prediction. These results show that network topology is an important factor in optimal network predictive performance.

The second set of controls shows that network design partially overcomes the multiple minima problem in the space of the network variables. For this purpose we considered a simpler version of network A, consisting of only the helix propensity [Fig. 3(b)], sheet propensity [Fig. 4(b)], sheet hydrophobicity for $(i-1,i,i+1)$ and $(i-2,i,i+2)$ [Fig. 5(b)], and helix hydrophobicity [Fig. 6(b) only]; this reduced subfunction space was fed directly into the intended output unit (helix hydrophobicity into helix output, etc.). When this simpler topology was started with completely randomized network variables and then trained, examination of the weights revealed that the intended purpose of the network was reproduced, i.e.,

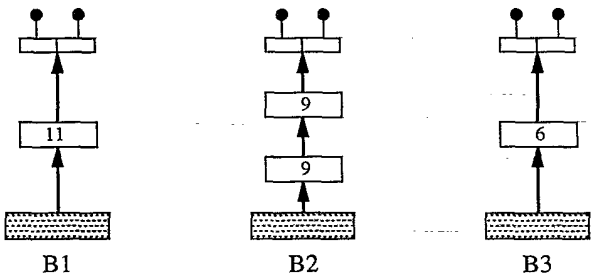

FIG. 9. Simple two-layer and three-layer networks. The B1 network fully connects the input layer to eleven hidden units that are connected to the two-state output (9__11__2 network). B2 refers to a 9__9__9__2 network, and B3 to a 9__6__2 network.

many of the individual topologies generally reproduced the original hand-designed functionality. For example, the logic of the helix propensity function was rather neatly preserved. If at least one of the lower subfunction outputs was helix promoting, the output of the overall helix propensity function was positive. A similar reproduction of functionality occurred, to various degrees, in the sheet propensity, sheet $(i,i+1)$ hydrophobicity, and sheet $(i,i+2)$ hydrophobicity topologies. However, the many-layered topologies, as in the helix hydrophobicity, were difficult to deconvolute into logical subfunctions, due to the problems of backpropagation through deeper networks. The excessive dilution of error signals toward the bottom of the network resulted in the helix hydrophobicity topology being the least functionally useful of all the trained topologies, its output weight an order of magnitude smaller than the others. Nonetheless, the remaining network did settle into a weight configuration mimicking the functionality of our intended network functions, and supports our assertion that the actual network topology is an important factor in controlling the "global" optimization of network variables.

The final set of controls illustrates how the hand-designed networks might themselves be optimized and understood. The key unknowns in our network topologies were the design of the hidden context units and hidden units used to integrate the hand-designed functions. Intuitively, we expected the hidden units in the latter category to combine hand-designed function outputs into a "known" set of sequence-structure correlations, and the latter to extract any remaining correlations that the hand-designed units did not explicitly encode. In actuality, the hand-designed topologies served as a template for how sequence-structure mappings are defined, and allowed the more flexible context hidden units to both learn from, and marginally improve performance of, the hand-designed pieces alone. In effect, the designed portion of the network served as constraints in a constrained optimization of the context hidden neurons. This is shown as follows.

We examined the effect of removing all hand-designed functions, and their associated hidden layer, from network A (Fig. 10) and recalculated only the output thresholds. The remaining network (A2, Table I) performed surprisingly well on both train and test sets, better than topologically equivalent 9__2__2 and optimal 9__6__2 networks. The key difference seemed to be that the hidden context neurons in network A were trained in tandem with the hand-designed functions (and their associated hidden layer).

Examination of the context neuron weights from the trained network A revealed that these hidden units were primarily using information from the helix and sheet propensity input fields. Next, a new network, A3 (Table I), was defined where the biases and the weights leading into the hidden units were fixed to their trained values of network A—effectively freezing the units into now learned "feature detectors." In addition, the hidden-to-output weights of the feature detectors were free to change, and two additional context hidden neurons, fully connected to input and output, were added (Fig. 10). Examination of

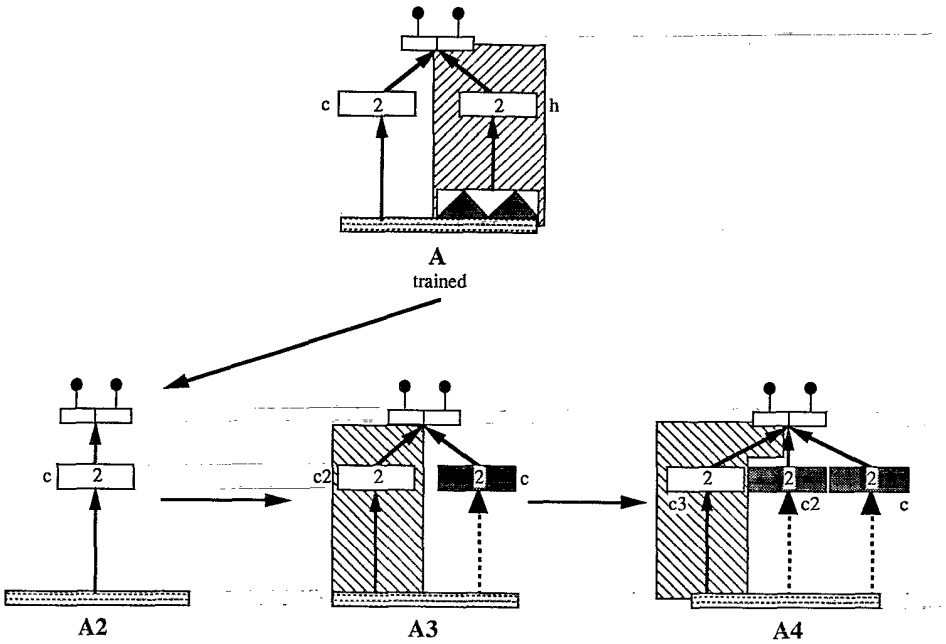

FIG. 10. The hand-designed network A (see Fig. 8) serves as a constraint on the optimization of the context network variables. A2 corresponds to a network with just the context neurons (c) trained in the presence of the hand-designed piece. The context neurons from A2 are frozen in network A3, and two additional context neurons are trained ($c2$). A4 is a network with context neurons (c) and ($c2$) frozen as feature detectors, and two additional context neurons ($c3$) are added.

the new hidden unit weights after training revealed that the second pair of context units were heavily coupled to the hydrophobicity field. Overall, this constraint-optimized network performed slightly better than network A, our optimal hand-designed based network, in both train and test sets.

We then added the learned hydrophobic correlation context neurons to those exploiting helix and sheet propensity, again fixing the lower portions of the new context neurons, and added yet another two hidden neurons (network A4, Fig. 10, Table I). After training, the added context neurons showed no favor for any particular portion(s) of the input representation (magnitudes of all weights were on the same order), unlike the trained hidden neurons in network A5 and network A. In fact, while the train set performance increased, the test set performance decreased significantly, which implies that the network was no longer extracting generalizations from the database. The above observations show that the parallel training of the hand-designed and context portions of the network produce the best network performance. Though there may be overlap in the knowledge learned by the two portions, the mutual error feedback is important in the training phase. It is interesting to note that our best hand-designed network A (Fig. 8) and learned network A3 (Fig. 10) seem to be extracting higher-order correlations than the simpler B networks; further testing of this assertion, such as analyzing the network weights and/or comparison to statistical methods with the same size input window, is required.

V. CONCLUSIONS

In summary, network topologies were individually hand designed to incorporate information on helix and sheet propensities, as well as the influence of individual amino acid hydrophobicities, to serve as constraints for a more optimal network solution to secondary-structure prediction. The primary benefit of the networks presented here is generalization superior to the test set of proteins as compared to randomly chosen topologies. While overall network performance was less than past neural-network efforts, we have not yet fully exploited input representation or window size in conjunction with our network architectures, which we hope to do in the near future.

The problems of database sparsity, deficient network architectures, and poor network variable optimization strategies are intimately coupled, and disentangling them is the primary direction we are pursuing in our neural-network approaches to protein-structure prediction. While it is necessary to further improve on network architecture (expansion of window size nd optimal input and output representations), this work clearly demonstrates that network design can positively impact generalization and network variable optimization. There is no question that database deficiencies remain a daunting problem for neural-network approaches to protein-secondary-structure prediction [34], although some compensation is always found by exploiting sequence and/or structural homologies. However, the results presented here hint at the possibility that network design might allow for more efficient mining of rules about complex sequence-structure relationships in the existing database, although further testing of this possibility is required. Consideration of model chemistries where *complete* sequence and structure databases can be defined might lend insight into the true severity of this problem and strategies for overcoming it [22,29]

ACKNOWLEDGMENTS

We gratefully acknowledge support from the U.S. Air Force Office of Sponsored Research, Grant No. F49620-94-1-0081, and the Office of Health and Environmental Research, U.S. Department of Energy, under Contract No. DE-AC03-76-SF00098.

[1] G. D. Fasman, in *Prediction of Protein Structure and the Principles of Protein Conformation,* edited by G. D. Fasman (Plenum, New York, 1989), Vol. 6, pp. 193–316.
[2] J. Garnier, Biochimie **72**, 513 (1990).
[3] J. D. Hirst and M. J. E. Sternberg, Biochemistry **31**, 7211 (1992).
[4] P. Y. Chou and G. D. Fasman, Biochemistry **13**, 222 (1974).
[5] J. Garnier, D. J. Osguthorpe, and B. Robson, J. Mol. Biol. **120**, 97 (1978).
[6] J. F. Gibrat, J. Garnier, and B. Robson, J. Mol. Biol. **198**, 425 (1987).
[7] O. B. Ptitsyn and A. V. Finkelstein, Protein Eng. **2**, 443 (1989).
[8] N. Qian and T. J. Sejnowski, J. Mol. Biol. **202**, 865 (1988).
[9] H. Bohr, J. Bohr, S. Brunak, R. M. J. Cotterill, B. Lautrup, L. Norskov, O. H. Olsen, and S. B. Petersen, FEBS Lett. **241**, 223 (1988).
[10] L. H. Holley and M. Karplus, Proc. Natl. Acad. Sci. USA **86**, 152 (1989).
[11] B. Rost, C. Sander, and R. Schneider, J. Mol. Biol. **235**, 13 (1994).
[12] D. G. Kneller, F. E. Cohen, and R. Langridge, J. Mol. Biol. **214**, 171 (1990).
[13] H. Bohr, J. Bohr, S. Brunak, R. M. J. Cotterill, H. Fredholm, B. Lautrup, and S. B. Petersen, FEBS Lett. **261**, 43 (1990).
[14] L. H. Holley and M. Karplus, Methods Enzymol. **202**, 204 (1991).
[15] S. Hayward and J. F. Collins, Protein: Struct. Funct. Genet. **14**, 372 (1992).
[16] S. M. Muskal and S.-H. Kim, J. Mol. Biol. **225**, 713 (1992).
[17] P. Stolorz, A. Lapedes, and Y. Xia, J. Mol. Biol. **225**, 363 (1992).
[18] X. Zhang, J. Mesirov, and D. Waltz, J. Mol. Biol. **225**, 1049 (1992).
[19] S. R. Holbrook, I. Dubchak, and S.-H. Kim, Biotechniques **14**, 984 (1993).
[20] P. Fariselli, M. Compiani, and R. Casadio, Eur. Biophys. J. **22**, 41 (1993).
[21] T. Head-Gordon and F. H. Stillinger, Biopolymers **33**, 293 (1993).
[22] T. Head-Gordon and F. H. Stillinger, Phys. Rev. E **48**, 1502 (1993).
[23] J. Greer, Proteins **7**, 317 (1990).
[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Nature **323**, 533 (1986).
[25] E. M. Johansson, F. U. Dowla, and D. M. Goodman, Int. J. Neural Syst. **2**, 291 (1992).
[26] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, Science **220**, 671 (1983).
[27] D. J. Montana and L. Davis, in *Eleventh International Joint Conference on Artificial Intelligence,* Detroit 1989, edited by N. Sridharan (Morgan Kaufman, San Mateo, CA, 1989), pp. 762–767.
[28] M. Levitt and C. Chothia, Nature **261**, 552 (1976).
[29] F. H. Stillinger, T. Head-Gordon, and C. L. Hirshfeld, Phys. Rev. E **48**, 1469 (1993).
[30] J. A. Hertz, R. G. Palmer, and A. S. Krogh, *Introduction to the Theory of Neural Computation* (Addison-Wesley, Redwood City, CA, 1991).
[31] W. Kabsch and C. Sander, FEBS Lett. **155**, 179 (1983).
[32] W. Kabsch and C. Sander, Biopolymers **22**, 257 (1983).
[33] M. Levitt, Biochemistry **17**, 4277 (1978).
[34] M. J. Rooman and S. J. Wodak, Nature **335**, 45 (1988).